TITLE: **Automatic Formatting and Validating of Text for a Markup Language Graphical User Interface**

INVENTOR: **Panagiotis Kougiouris and Chip Bering**

APPENDIX B

Appendix B

```
#include <olectl.h>
// hsDHTMLControl.idl : IDL source for hsDHTMLControl.dll
//

// This file will be processed by the MIDL tool to
// produce the type library (hsDHTMLControl.tlb) and marshalling code.

import "oaidl.idl";
import "ocidl.idl";

#include <mshtmdid.h>

    [
        object,
        uuid(17F34ED4-FB59-11D1-801A-00201829472A),
        dual,
        helpstring("IHSDHTMLControl Interface"),
        pointer_default(unique)
    ]
    interface IHSDHTMLControl : IDispatch
    {

        // Error codes come form this enumerations.
        // The severity is SEVERITY_ERROR and the facility is FACILITY_ITF
        typedef enum HSFTDHTMLControlError {
            E_IHSDC_ERR_NULL_DOCUMENT_OBJECT = 0x201,
            E_IHSDC_ERR_NO_RDS_CONTROL = 0x202,
            E_IHSDC_ERR_OBJECT_IS_NO_PROPER_INTERFACE = 0x203,
            E_IHSDC_ERR_NO_PARSED_PAGE = 0x204,
            E_IHSDC_ERR_UNKNOWN_HSTYPE = 0x205,
            E_IHSDC_ERR_UNKNOWN_DHTMLCONTROL = 0x206,
            E_IHSDC_ERR_NO_SUCH_OBJECT = 0x207,
            E_IHSDC_ERR_RECORDSET_EXPECTED = 0x208,
            E_IHSDC_ERR_NOT_SUPPORTED_ELEMENT = 0x209
        } HSFTDHTMLControlError;


        // **** HTML page convetnions
        // HTML pages loaded to this control are expected to follow the conventions:
        //
        // They should contain the following  object
        // <object id="hsDHTMLCtl" classid="clsid:17F34ED5-FB59-11D1-801A-00201829472A"
        // align="baseline" border="0" width="0" height="0">


        // This method should be called every time the layout of a page changes (e.g. naviga
te to
        // a different document, add remove controls etc.)
        [id(12), helpstring("method ParseHTMLPage")] HRESULT ParseHTMLPage();


        // A helper function that creates a recordset based on the elements in the form.
        // Only elements containing the attribute DATASRC (or HSDATASRC) are treated by the
control.
        // Attributes with HSTYPE are put in the recordset (field DATAFLD or HSDATAFLD).
        // Attributes with HSFORMAT are validated.
        // The type of the field is based on the value of the attribute.
        // The name of the field is based on the "datafld" attribute (or HSDATAFLD).
        // hSTYPE is used for the recordset, HSFORMAT is used for validation
        // Here are supported:
```

```
//
// HSTYPE
// ------
// string      -- implemented
// boolean     -- implemented
// integer     -- implemented
// date        -- implemented
// smallInt    -- implemented
// currency    -- implemented
// double      -- implemented
//
// HSFORMAT
// --------
        // boolean
        //
        //      Valid forms;    "1" or "0".
        //      External form;  "1" or "0".
        //      Internal form;  "1" or "0"
        //
        // cobcode
        //
        //      Valid forms;    X Where 'X' denotes a letter or a digit.
        //                      There must be only 1 character.
        //      External form;  X Where 'X' denotes a letter or a digit.
        //      Internal form;  X Where 'X' denotes a letter or a digit.
        //
        // cpt4code
        //
        //      Valid forms;    99999 Where '9' denotes a digit.
        //                      There must be 5 digits.
        //      External form;  99999 Where '9' denotes a digit.
        //      Internal form;  99999 Where '9' denotes a digit.
        //
        // date
        //
        //      Valid forms;    Any of the date forms shown in the "Regional
        //                      settings of the control panel.
        //      External form;  The date format currently set for the local machine.
        //                      This is set in the control panel under "Regional Set
tings".  When the regional setting does not show four digit years, the setting is modified -
 only for purposes of the date control - to use four digit years.
        //      Internal form;  mm/dd/yyyy This is the standard Informix date format
.
        //
        // datetime
        //
        //      Valid forms;    Any of the date and time forms shown in the
        //                      "Regional settings of the control panel.
        //       External form; The date and time formats currently set for the loca
l
        //                      machine.   This is set in the control panel under
        //                      "Regional Settings".   When the regional setting doe
s
        //                      not show four digit years, the setting
        //                      is modified - only for purposes of the date control
        //                      - to use four digit years.
        //      Internal form;  mm/dd/yyyy hh:mm:ss  This is standard Informix
        //                      date time format.
        //
        //  hcpcscode
```

```
//
//      Valid forms;    A9999 Where A is a character A thru V (inclusive).
//                      The characters may be upper or lower case.  There
//                      must be 5 characters.
//      External form;  A9999.  The first character is converted to uppercas
e.
//      Internal form;  A9999
//
//  icdcode
//
//      Valid forms;    a.b Where 'a' denotes 1-3 digits or a letter followe
d
//                      by 2 digits, and b denotes 1-2 digits.  The decimal
//                      point is optional if there are no digits after the d
ecimal.
//      External form;  a.b Where 'a' denotes 1-3 digits or a letter followe
d by
//                      2 digits, and b denotes 1-2 digits.  The decimal poi
nt
//                      is optional if there are no digits after the decimal
.
//      Internal form;  a.b Where 'a' denotes 1-3 digits or a letter followe
d
//                      by 2 digits, and b denotes 1-2 digits.  The decimal
//                      point is optional if there are no digits after the d
ecimal.
//
//  integer
//
//      Valid forms;    integer values.  Negative signs allowed as either
//                      "-" or the value in parenthesis.
//      External form;  integer value with negative indicated as per the
//                      regional settings.
//      Internal form;  integer value with "-" to indicate negatives.
//
//  Name
//
//      Valid forms;    Any text.
//      External form;  The validator removes all leading and trailing space
s
//                      and then replaces runs of multiple whitespace with a
//                      single blank character.  The following patterns are
handled;
//
//
//          Pattern                 Translated to
//          Token1, token2, token3  Token1, token2, token3
//          Token1, Token2          Token1, Token2
//          Token                   Token
//          Token1 Token2               Token2, Token1
//          Token1 Token2 Token3    Token3, Token1, Token2
//
//                  Words that are of a single case (all lowercase character
s
//                      or all uppercase characters) are translated to prope
r
//                      form where the first character is uppercase and the
//                      others are lowercase.
//       Internal form; Same as external form.
```

```
//
//  searchable
//
//      Valid forms;    Any string that does not contain one of the characte
rs
//                      "?*[]%". This is used for fields that would be used
//                      for a search.
//      External form;  The string.
//      Internal form;  The string.
//
//  smallint
//
//      Valid forms;    integer values in the range SHRT_MIN to SHRT_MAX (in
clusie).
//                      Negative signs allowed as either
//                      "-" or the value in parenthesis.
//      External form;  integer value with negative indicated as per the
//                      regional settings.
//      Internal form;  integer value with "-" to indicate negatives.
//
//  time
//
//      Valid forms;    Any of the time forms shown in the "Regional
//                      settings of the control panel.
//      External form;  The time format currently set for the local machine.

//                      This is set in the control panel under "Regional Set
tings".
//      Internal form;  hh:mm:ss This is standard Informix time format.
//
//  usein
//
//      Valid forms;    999-99-9999 Where '9' denotes a digit.
//                      Dashes are optional.
//      External form;  999-99-9999 Where '9' denotes a digit.
//      Internal form;  999999999 Where '9' denotes a digit.
//
//              Note this is identical to SSN.
//
// usmoney
//
//      Valid forms;    Any of the currency forms shown in the "Regional
//                      settings of the control panel.
//      External form;  The currency format currently set for the local mach
ine.
//                      This is set in the control panel under "Regional Set
tings".
//      Internal form;  A floating point number.
//
//  usphone
//
//      Valid forms;    (999)999-9999 Where '9' denotes a digit, or 999.99.9
999.
//                      Everything but the digits are optional.  The validat
or
//                      is also very flexible about accepting intervening
//                      whitespaces.
//      External form;  (999) 999-9999 Where '9' denotes a digit.
//      Internal form;  9999999999 Where '9' denotes a digit.
//
```

```
                //  usssn
                //
                //      Valid forms;    999-99-9999 Where '9' denotes a digit.  Dashes are o
ptional.
                //      External form;  999-99-9999 Where '9' denotes a digit.
                //      Internal form:  999999999 Where '9' denotes a digit.
                //
                //  Note that this is identical to EIN.
                //
                //  usState
                //
                //      Valid forms;    Any one of the following;
                //
                //              "AK","AL","AR","AZ","CA","CO","CT","DC","DE","FL","GA","HI",
                //              "IA","ID","IL","IN","KS","KY","LA","MA","MD","ME","MI","MN",
                //              "MO","MS","MT","NC","ND","NE","NH","NJ","NM","NV","NY","OH",
                //              "OK","OR","PA","PR","RI","SC","SD","TN","TX","UT","VA","VI",
                //              "VT","WA","WI","WV","WY".
                //
                //      External form;  The validator removes all leading and trailing space
s,
                //                      compares against the valid forms, and changes the ca
se
                //                      to upper case.
                //      Internal form:  Same as external form.
                //
                // usStreet
                //
                //      Valid forms;    Any text.
                //      External form;  The validator removes all leading and trailing space
s
                //                      and then replaces runs of multiple whitespace with a
                //                      single blank character.  The first letters of words,
                //                      which are runs of letters seperated by blanks, are s
et to
                //                      uppercase and all the following letters are set to
                //                      lowercase if all of the letters in the word are a si
ngle
                //                       case (all upper or all lower).
                //      Internal form;  Same as external.
                //
                //  yesno
                //
                //      Valid forms;    "Y", "y" "Yes", "YEs", "YES", "YeS", "N", "n", "No",
                //                      "NO", "nO", "no".
                //      External form;  "Yes" or "No".  Leading and trailing whitespace is r
emoved.
                //      Internal form:  Same as external.
                //
                //  usZipcode
                //
                //      Valid forms;    "99999" or "99999-9999" where '9' denotes a digit.
                //      External form;  "99999".  The zip+4 is removed.
                //      Internal form;  "99999" where 9 denotes a digit.
        //
        // implemented
        [id(4), helpstring("method CreateListOfHTMLElements")] HRESULT CreateHTMLElements([i
```

```
n]BSTR dataSource, [out, retval] LPDISPATCH* ADORRecordset);

        // Get and set the recordset that maps to the HTML Elements. The returned recordsets
 are typed.
        // STATUS: implemented
        [propget, id(11), helpstring("property HTMLElements")] HRESULT HTMLElements([in]BSTR
 dataSource, [out, retval] LPDISPATCH *pVal);
        [propputref, id(11), helpstring("property HTMLElements")] HRESULT HTMLElements([in]B
STR dataSource, [in] LPDISPATCH newVal);

        // The untyped counterparts of the above functions
        // The returned recordsets are untyped (HSTYPE ignored and we return strings).
        [id(24), helpstring("method CreateListOfUntypedHTMLElements")] HRESULT CreateUntyped
HTMLElements([in]BSTR dataSource, [out, retval] LPDISPATCH* ADORRecordset);
        [id(25), helpstring("the reverse of the parse method, cleans up resources")] HRESULT
 Cleanup();

        // Get and set the recordset that maps to the HTML Elements.
        // STATUS: implemented
        [propget, id(21), helpstring("property UntypedHTMLElements")] HRESULT UntypedHTMLEle
ments([in]BSTR dataSource, [out, retval] LPDISPATCH *pVal);
        [propputref, id(21), helpstring("property UntypedHTMLElements")] HRESULT UntypedHTML
Elements([in]BSTR dataSource, [in] LPDISPATCH newVal);

        // This property can be set using:
        // <PARAM NAME="ValidateOnKeyUp" VALUE="0">
        [propget, id(13), helpstring("property ValidateOnKeyUp")] HRESULT ValidateOnKeyUp([o
ut, retval] BOOL *pVal);
        [propput, id(13), helpstring("property ValidateOnKeyUp")] HRESULT ValidateOnKeyUp([i
n] BOOL newVal);

        // This property can be set using:
        // <PARAM NAME="InvalidClassName" VALUE="INVALID">
        [propget, id(14), helpstring("property InvlalidClass")] HRESULT InvalidClassName([ou
t, retval] BSTR *pVal);
        [propput, id(14), helpstring("property InvlalidClass")] HRESULT InvalidClassName([in
] BSTR newVal);

        // After setting the recordset all the elements are valid. As the user changes value
s the
        // controls raises the event and asks if this is a valid change. If any listener ind
icates
        // that this is not a valid change it is marked as not valid and the recordset is no
t updated.
        [propget, id(3), helpstring("property AreAllElementsValid")] HRESULT AreAllHTMLEleme
ntsValid([in]BSTR dataSource, [out, retval] BOOL *pVal);

        // An alternative of the AreAllHTMLElementsValid() methods that also returns a safea
rray with the
        // ids od the invalid elements. If the array has 0 elements all the elements are val
id
        [id(30), helpstring("method InvalidElements")] HRESULT InvalidElements([in]BSTR data
Source, [out]SAFEARRAY(BSTR)* dataSourceElems);

        // When the SetHTMLElement method is called we clear or not the elements based
        // on the value of this property
        //
        // In 2.0 default is OFF
        // This will change in future releases
        // This property can be set using:
```

```
        // <PARAM NAME="ResetElementContentsOnSet" VALUE="1">
        [propget, id(31), helpstring("property ResetElementContentsOnSet")] HRESULT ResetEle
mentContentsOnSet([out, retval] BOOL *pVal);
        [propput, id(31), helpstring("property ResetElementContentsOnSet")] HRESULT ResetEle
mentContentsOnSet([in] BOOL newVal);

        // This is used to set elements like listboxes and comboboxes that cannot be set dir
ectly
        // from data binding. It is set before setting the HTMLElements property
        [id(5), helpstring("method SetHTMLElement")] HRESULT SetHTMLElement([in]BSTR idOfEle
ment, [in]SAFEARRAY(VARIANT)* values);

        // Same as above but uses a column in a recordset to get values
        [id(6), helpstring("method SetHTMLElement")] HRESULT SetHTMLElementFromRecordset([in
]BSTR idOfElement, [in]LPDISPATCH adorRecordset, [in]BSTR fieldName);

        // Reset controls by clearing all text. Doesn't delete controls, just resets content
s
        [id(32), helpstring("Clear All HTML Element Display Text")] HRESULT ClearAllHTMLElem
entDisplayText();

        // Provide folks a way of determining/getting "dirty" html elements (BUGno24507)
        [propget, id(33), helpstring("property (read only) DirtyElementsExist flag")]
            HRESULT DirtyElementsExist([in]BSTR dataSource, [out, retval] VARIANT_BOOL *pVal
);
        [id(34), helpstring("Clear any elements that are marked dirty.")]
            HRESULT ClearDirtyFlags([in]BSTR dataSource);
        [id(35), helpstring("Get the list of modified elements. Returns element identifiers"
)]
            HRESULT GetDirtyElementIDs([in]BSTR dataSource, [out] SAFEARRAY(BSTR)* elementID
s);

        // XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        // Deprectaed methods please do not use
        // XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        // Get the containing browser. From it you get go to document, window, catch events
etc.
        // This method should not be necessary any more
        // implemented
        [propget, id(1), helpstring("property Browser")] HRESULT Browser([out, retval] LPDIS
PATCH* pVal);

        // This is a tricky method. The major porblem is to time when to call it. If you set
 the recordset and
        // immediately call it, it wont work, because the population of the widgets from the
 recordset
        // happens asynchronously and is not clear to me when we are done
        [id(7), helpstring("do the validation")]HRESULT Validate([in]BSTR dataSource);

    };



    [
    hidden,
    dual,
    object,
    uuid(3050f33c-98b5-11cf-bb82-00aa00bdce0b)
    ]
    interface HTMLElementEvents : IDispatch
```

```
    {
        [id(DISPID_HTMLELEMENTEVENTS_ONHELP)]  HRESULT onhelp([out, retval]VARIANT_BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONCLICK)]  HRESULT  onclick([out, retval]VARIANT_BOOL*)
;
        [id(DISPID_HTMLELEMENTEVENTS_ONDBLCLICK)]  HRESULT  ondblclick([out, retval]VARIANT_
BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONKEYPRESS)]  HRESULT  onkeypress([out, retval]VARIANT_
BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONKEYDOWN)] HRESULT  onkeydown();
        [id(DISPID_HTMLELEMENTEVENTS_ONKEYUP)] HRESULT  onkeyup();
        [id(DISPID_HTMLELEMENTEVENTS_ONMOUSEOUT)] HRESULT onmouseout();
        [id(DISPID_HTMLELEMENTEVENTS_ONMOUSEOVER)] HRESULT  onmouseover();
        [id(DISPID_HTMLELEMENTEVENTS_ONMOUSEMOVE)] HRESULT  onmousemove();
        [id(DISPID_HTMLELEMENTEVENTS_ONMOUSEDOWN)] HRESULT  onmousedown();
        [id(DISPID_HTMLELEMENTEVENTS_ONMOUSEUP)] HRESULT  onmouseup();
        [id(DISPID_HTMLELEMENTEVENTS_ONSELECTSTART)] HRESULT  onselectstart([out, retval]VAR
IANT_BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONFILTERCHANGE)] HRESULT  onfilterchange();
        [id(DISPID_HTMLELEMENTEVENTS_ONDRAGSTART)]  HRESULT ondragstart([out, retval]VARIANT
_BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONBEFOREUPDATE)] HRESULT  onbeforeupdate([out, retval]V
ARIANT_BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONAFTERUPDATE)] HRESULT  onafterupdate();
        [id(DISPID_HTMLELEMENTEVENTS_ONERRORUPDATE)] HRESULT  onerrorupdate([out, retval]VAR
IANT_BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONROWEXIT)]  HRESULT onrowexit([out, retval]VARIANT_BOO
L*);
        [id(DISPID_HTMLELEMENTEVENTS_ONROWENTER)] HRESULT  onrowenter();
        [id(DISPID_HTMLELEMENTEVENTS_ONDATASETCHANGED)] HRESULT  ondatasetchanged();
        [id(DISPID_HTMLELEMENTEVENTS_ONDATAAVAILABLE)] HRESULT  ondataavailable();
        [id(DISPID_HTMLELEMENTEVENTS_ONDATASETCOMPLETE)] HRESULT  ondatasetcomplete();
    };

    [
    hidden,
    dual,
    object,
    uuid(3050f2a7-98b5-11cf-bb82-00aa00bdce0b)
    ]
    interface HTMLInputTextElementEvents : IDispatch
    {
        [id(DISPID_HTMLELEMENTEVENTS_ONHELP)]  HRESULT onhelp([out, retval]VARIANT_BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONCLICK)]  HRESULT  onclick([out, retval]VARIANT_BOOL*)
;
        [id(DISPID_HTMLELEMENTEVENTS_ONDBLCLICK)]  HRESULT  ondblclick([out, retval]VARIANT_
BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONKEYPRESS)]  HRESULT  onkeypress([out, retval]VARIANT_
BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONKEYDOWN)] HRESULT  onkeydown();
        [id(DISPID_HTMLELEMENTEVENTS_ONKEYUP)] HRESULT  onkeyup();
        [id(DISPID_HTMLELEMENTEVENTS_ONMOUSEOUT)] HRESULT onmouseout();
        [id(DISPID_HTMLELEMENTEVENTS_ONMOUSEOVER)] HRESULT  onmouseover();
        [id(DISPID_HTMLELEMENTEVENTS_ONMOUSEMOVE)] HRESULT  onmousemove();
        [id(DISPID_HTMLELEMENTEVENTS_ONMOUSEDOWN)] HRESULT  onmousedown();
        [id(DISPID_HTMLELEMENTEVENTS_ONMOUSEUP)] HRESULT  onmouseup();
        [id(DISPID_HTMLELEMENTEVENTS_ONSELECTSTART)] HRESULT  onselectstart([out, retval]VAR
IANT_BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONFILTERCHANGE)] HRESULT  onfilterchange();
        [id(DISPID_HTMLELEMENTEVENTS_ONDRAGSTART)]  HRESULT ondragstart([out, retval]VARIANT
_BOOL*);
```

```
        [id(DISPID_HTMLELEMENTEVENTS_ONBEFOREUPDATE)] HRESULT  onbeforeupdate([out, retval]V
ARIANT_BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONAFTERUPDATE)] HRESULT  onafterupdate();
        [id(DISPID_HTMLELEMENTEVENTS_ONERRORUPDATE)] HRESULT  onerrorupdate([out, retval]VAR
IANT_BOOL*);
        [id(DISPID_HTMLELEMENTEVENTS_ONROWEXIT)]  HRESULT onrowexit([out, retval]VARIANT_BOO
L*);
        [id(DISPID_HTMLELEMENTEVENTS_ONROWENTER)] HRESULT  onrowenter();
        [id(DISPID_HTMLELEMENTEVENTS_ONDATASETCHANGED)] HRESULT  ondatasetchanged();
        [id(DISPID_HTMLELEMENTEVENTS_ONDATAAVAILABLE)] HRESULT  ondataavailable();
        [id(DISPID_HTMLELEMENTEVENTS_ONDATASETCOMPLETE)] HRESULT  ondatasetcomplete();
        [id(DISPID_HTMLCONTROLELEMENTEVENTS_ONFOCUS)] HRESULT onfocus();
        [id(DISPID_HTMLCONTROLELEMENTEVENTS_ONBLUR)] HRESULT onblur();
        [id(DISPID_HTMLCONTROLELEMENTEVENTS_ONRESIZE)] HRESULT onresize();
        [id(DISPID_HTMLINPUTTEXTELEMENTEVENTS_ONCHANGE)] HRESULT onchange([out, retval]VARIA
NT_BOOL*);
        [id(DISPID_HTMLINPUTTEXTELEMENTEVENTS_ONSELECT)] HRESULT onselect();
    };


    [
      uuid(34A715A0-6587-11D0-924A-0020AFC7AC4D),
      helpstring("Web Browser Control events interface"),
      dual,
      hidden
    ]
    interface DWebBrowserEvents2 : IDispatch
    {
            [id(0x00000066), helpstring("Statusbar text changed.")]
            HRESULT StatusTextChange([in] BSTR Text);
            [id(0x0000006c), helpstring("Fired when download progress is updated.")]
            HRESULT ProgressChange(
                        [in] long Progress,
                        [in] long ProgressMax);
            [id(0x00000069), helpstring("The enabled state of a command changed.")]
            HRESULT CommandStateChange(
                        [in] long Command,
                        [in] VARIANT_BOOL Enable);
            [id(0x0000006a), helpstring("Download of a page started.")]
            HRESULT DownloadBegin();
            [id(0x00000068), helpstring("Download of page complete.")]
            HRESULT DownloadComplete();
            [id(0x00000071), helpstring("Document title changed.")]
            HRESULT TitleChange([in] BSTR Text);
            [id(0x00000070), helpstring("Fired when the PutProperty method has been called."
)]
            HRESULT PropertyChange([in] BSTR szProperty);
            [id(0x000000fa), helpstring("Fired before navigate occurs in the given WebBrowse
r (window or frameset element). The processing of this navigation may be modified.")]
            HRESULT BeforeNavigate2(
                        [in] IDispatch* pDisp,
                        [in] VARIANT* URL,
                        [in] VARIANT* Flags,
                        [in] VARIANT* TargetFrameName,
                        [in] VARIANT* PostData,
                        [in] VARIANT* Headers,
                        [out] VARIANT_BOOL* Cancel);
            [id(0x000000fb), helpstring("A new, hidden, non-navigated WebBrowser window is n
eeded.")]
            HRESULT NewWindow2(
```

```
                        [out] IDispatch** ppDisp,
                        [out] VARIANT_BOOL* Cancel);
            [id(0x000000fc), helpstring("Fired when the document being navigated to becomes
visible and enters the navigation stack.")]
            HRESULT NavigateComplete2(
                        [in] IDispatch* pDisp,
                        [in] VARIANT* URL);
            [id(0x00000103), helpstring("Fired when the document being navigated to reaches
ReadyState_Complete.")]
            HRESULT DocumentComplete(
                        [in] IDispatch* pDisp,
                        [in] VARIANT* URL);
            [id(0x000000fd), helpstring("Fired when application is quiting.")]
            HRESULT OnQuit();
            [id(0x000000fe), helpstring("Fired when the window should be shown/hidden")]
            HRESULT OnVisible([in] VARIANT_BOOL Visible);
            [id(0x000000ff), helpstring("Fired when the toolbar  should be shown/hidden")]
            HRESULT OnToolBar([in] VARIANT_BOOL ToolBar);
            [id(0x00000100), helpstring("Fired when the menubar should be shown/hidden")]
            HRESULT OnMenuBar([in] VARIANT_BOOL MenuBar);
            [id(0x00000101), helpstring("Fired when the statusbar should be shown/hidden")]
            HRESULT OnStatusBar([in] VARIANT_BOOL StatusBar);
            [id(0x00000102), helpstring("Fired when fullscreen mode should be on/off")]
            HRESULT OnFullScreen([in] VARIANT_BOOL FullScreen);
            [id(0x00000104), helpstring("Fired when theater mode should be on/off")]
            HRESULT OnTheaterMode([in] VARIANT_BOOL TheaterMode);
    };

    // ************************************************************************
    // The validation stuff
    // ************************************************************************
    [
        object,
        uuid(6AD8D484-0490-11d2-801D-00201829472A),
        dual,
        helpstring("IHSFormatter Interface"),
        pointer_default(unique)
    ]
    interface IHSFormatter : IDispatch
    {
        typedef enum HSFTFormat {
            HSDTDefault = 0,    // usually regional settings
            HSDTHealtheon = 1   // the Healtheon representations
        } HSFTFormat;

        // The Formatter parses the string to an interna format.
        // This property specifies how this parsing is done
        [propget, id(13), helpstring("property inFormat")] HRESULT InFormat([out, retval] HS
FTFormat* pVal);
        [propput, id(13), helpstring("property inFormat")] HRESULT InFormat([in] HSFTFormat
newVal);

        // The Formatter outputs the interna format to a string
        // This property specifies how this parsing is done
        [propget, id(14), helpstring("property outFormat")] HRESULT OutFormat([out, retval]
HSFTFormat* pVal);
        [propput, id(14), helpstring("property outFormat")] HRESULT OutFormat([in] HSFTForma
t newVal);

        // This method takes the input string, parses it to an internal representation (usin
```

```
g inFormat prop)
        // then outputs to outArg using the outFormat prop
        // If the string is invalid the result is undetermined
        [id(2), helpstring("method GetConforming")] HRESULT GetConforming([in]BSTR inArg, [o
ut, retval]BSTR* outArg);

        // An the string be parsed? IsParseable could be a better name
        // If it is invalid it returns false and sets the reason. Otherwise returns true. No
n conforming
        // strings can be valid e.g.:
        // "408 345-6578 " is a valid us telephone number
        // The inFormat prop is used for determining if it is valid
        [id(1), helpstring("method IsValid")] HRESULT IsValid([in]BSTR argString, [out]BSTR*
 reason, [out, retval]VARIANT_BOOL* retVal);

        // BUGno16461 -- GetConformingIfValid() removed (no longer required)
        //[id(3), helpstring("method GetConformingIfValid")] HRESULT GetConformingIfValid([i
n]BSTR argString,
        //      [out]BSTR* reason, [out]BSTR* strConformingValue, [out, retval]VARIANT_BOOL*
 isValid);

        //************************************************
        // Deprecated methods DO NOT USE Removed beyond 2.0
        // Set the inFormat and OutFormat properties instead and use the other methods

        // This method returns a conforming string based on a format
        // that is passed in as an argument
        //[id(4), helpstring("Get a specific format")] HRESULT GetConformingWithFormat([in]H
SFTFormat fmt,
        //      [in]BSTR argString, [out, retval]BSTR* strConformingValue);

        // This method returns a conforming string based on a format
        // that is passed in as an argument
        //[id(5), helpstring("Get a specific format")] HRESULT GetConformingWithFormatIfVali
d([in]HSFTFormat fmt,
        //      [in]BSTR argString, [out]BSTR* reason, [out]BSTR* strConformingValue, [out,
retval]VARIANT_BOOL* isValid);
    };

    [
        object,
        uuid(3CA22A51-CAC3-11d2-A19B-00105A214053),
        dual,
        helpstring("IHSFormatter2 Interface -- implements FT 2.2 enhancements"),
        pointer_default(unique)
    ]
    interface IHSFormatter2 : IHSFormatter
    {
        // These two methods are for setting and clearing the list of params
        // for a given formatter. Clients of this interface should clear before
        // appending a list of format parameters.
        [id(15), helpstring("Clear parameter list")]
            HRESULT ClearFormatParamList();
        [id(16), helpstring("Append a (key/value) pair to list of params")]
            HRESULT AppendFormatParam([in] BSTR paramKey, [in] BSTR paramVal);
    };

    [
        object,
        uuid(E7047534-0404-11D2-801D-00201829472A),
```

```
        dual,
        helpstring("IHSValidator Interface"),
        pointer_default(unique)
    ]
    interface IHSValidator : IDispatch
    {
        [propget, bindable, defaultbind,  id(0), helpstring("property Formatter object")] HR
ESULT Formatter([in]BSTR, [out, retval] IHSFormatter **pVal);
    };


    [
        object,
        uuid(5C9B99E7-2D59-11D2-8E1C-00104B79DD7C),
        dual,
        helpstring("IHSComboBox Interface"),
        pointer_default(unique)
    ]
    interface IHSComboBox : IDispatch
    {
        [propput, id(DISPID_AUTOSIZE)]
        HRESULT AutoSize([in]VARIANT_BOOL vbool);
        [propget, id(DISPID_AUTOSIZE)]
        HRESULT AutoSize([out,retval]VARIANT_BOOL* pbool);
        [propput, id(DISPID_ENABLED)]
        HRESULT Enabled([in]VARIANT_BOOL vbool);
        [propget, id(DISPID_ENABLED)]
        HRESULT Enabled([out,retval]VARIANT_BOOL* pbool);
        [propput, id(DISPID_TEXT)]
        HRESULT Text([in]BSTR strText);
        [propget, id(DISPID_TEXT)]
        HRESULT Text([out, retval]BSTR* pstrText);
        [propput, id(DISPID_TABSTOP)]
        HRESULT TabStop([in]VARIANT_BOOL vbool);
        [propget, id(DISPID_TABSTOP)]
        HRESULT TabStop([out,retval]VARIANT_BOOL* pbool);
        [propget, id(1), helpstring("property Style")] HRESULT Style([out, retval] short *pV
al);
        [propput, id(1), helpstring("property Style")] HRESULT Style([in] short newVal);
        [propget, id(2), helpstring("property value")] HRESULT value([out, retval] VARIANT *
pVal);
        [propput, id(2), helpstring("property value")] HRESULT value([in] VARIANT newVal);
        [id(0xfffffdd7), helpstring("method AddItem")] HRESULT AddItem(
                        [in, optional] VARIANT pvargItem,
                        [in, optional] VARIANT pvargIndex);
        [propget, id(3), helpstring("property maxLength")] HRESULT maxLength([out, retval] s
hort *pVal);
        [propput, id(3), helpstring("property maxLength")] HRESULT maxLength([in] short newV
al);

        // Removed all the elements from the combo box
        [id(4), helpstring("method Clear")] HRESULT Clear();
    };
[
    uuid(17F34EA1-FB59-11D1-801A-00201829472A),
    version(1.0),
    helpstring("Healtheon DHTML Forms Filler Control 1.0 Type Library")
]
library HSDHTML
{
```

```
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");

    enum HSFTDHTMLControlError;
    enum HSFTFormat;

    //------------------------------------------------------------
    //Declare an interface for the  object's events
    // dispinterface: Event interfaces MUST be dispinterfaces if they are to be used by VB/S
cript
    //
    // hidden       : because there is a CoClass that implements this object as it's default
 event interface.
    //                The CoClass makes the object createable.  The when viewing the createa
ble object in the
    //                Object Browser displays the default interface
    //------------------------------------------------------------
    [
      uuid(B074923E-FB63-11d1-801A-00201829472A),
      hidden,
      nonextensible
    ]
       dispinterface IHSDHTMLControlEvents  {
        properties:
        methods:
            //-------------------------------------------------
            [id(10), helpstring("Event triggered when a property changes")]
            BOOL HTMLElelementBeforeUpdate([in] BSTR idOfElement, [in]VARIANT Value, [out]BS
TR* reason);
    }


    //-------------------------------------------------------------------------------------
    //
    //
    [
        uuid(17F34ED5-FB59-11D1-801A-00201829472A),
        helpstring("Healtheon DHTML Forms Filler Control")
    ]
    coclass Control
    {
        [default] interface IHSDHTMLControl;
        //-------------------------------------------------------
        //Interface of events we want to expose
        //-------------------------------------------------------
        [default, source] dispinterface IHSDHTMLControlEvents;
        interface DWebBrowserEvents2;
        interface HTMLElementEvents;
        interface HTMLInputTextElementEvents;
    };

    // This is the validator main object. It contains all the validator objects
    [
        uuid(E7047535-0404-11D2-801D-00201829472A),
        helpstring("Healtheon UI Validator")
    ]
    coclass HSValidator
    {
        [default] interface IHSValidator;
    };
```

```
    [
        uuid(443D4590-33B8-11d2-8E23-00104B79DD7C),
        version(1.0),
        hidden,
        nonextensible
    ]

    dispinterface  DIHSComboBoxEvents {
    properties:
    methods:
        [id(0x00000101)]void OnClick();
        [id(0x00000102)]void OnDropDown();
        [id(0x00000103)]void OnFocus();
        [id(0x00000104)]void OnBlur();
        [id(0x00000105)]void OnChange();
        [id(0x00000106)]void OnKeyDown(short pnChar, short nShiftState);

    };
    [
        uuid(5C9B99EA-2D59-11D2-8E1C-00104B79DD7C),
        helpstring("Healtheon HSComboBox Control")
    ]
    coclass HSComboBox
    {
        [default] interface IHSComboBox;
        [default, source] dispinterface DIHSComboBoxEvents;
    };
};
```